多段の数量推論タスクに対する適応的なモデルの振る舞いの検証

青木洋一 [1] 工藤慧音 [1] Ana Brassard[3,1] 栗林樹生 [1,2] 吉川将司 [1,3] 乾健太郎 [1,3]
[1] 東北大学 [2]Langsmith 株式会社 [3] 理化学研究所
{youichi.aoki.p2,keito.kudo.q4}@dc.tohoku.ac.jp,
ana.brassard@riken.jp, {kuribayashi,yoshikawa,inui}@tohoku.ac.jp

概要

数量推論において, Transformer [1] ベースのモデルは演算の複雑さが異なる問題に対しても常に固定の層の数で推論が行われるため, 演算をモデル内部で表現することが難しいと考えられる. 本稿では問題に応じて層の深さを適応的に変化させるモデルである PonderNet [2] が多段の推論を要する数量推論タスクに対してどのように作用するのかを検証した. その結果, 形式言語によって構成される数量推論データセットの中でも簡単な演算に対しての正解率は非適応的なモデルと適応的なモデルでは大きな差は見られなかったが, 適応的なモデルの層の深さは入力に応じて変化することが確認できた.

1 はじめに

深層学習ベースの自然言語処理システムの数量推論性能が向上することは科学や金融などの分野における応用において有益である一方, 数量推論に対する全体的な解決策はまだ見つかっていない [3]. 例えば, 算術演算を要する読解問題タスクである DROP [4] に対して, GenBERT [5] が高い精度を達成する一方で, FinQA [6] のような現実の問題に近い設定の場合, ニューラルネットワークは人間のパフォーマンスにははるかに及ばない事が分かっている. また, 学習時に見たことがない桁の数に対する操作や演算を行えない外挿 [7] の問題も有名である.

このような問題は, モデルの構造が数量推論に適していない事に起因すると本稿では仮説する. 一般的に使われている Transformer ベースのモデルに解かせる場合, 「$1+1$」や, 「922×312」のように演算の複雑さが異なる問題に対しても, 常に固定の層の数で推論が行われる. したがって, 問題に応じて層の深さが変化する事がないモデルは演算を内部で表現することが難しいと考えられる.

本稿では, 問題に応じて層の深さが適応的に変化するモデルとして, PonderNet [2] が多段の推論を要する数量推論タスクに対してどのように作用するのかを検証し, 非適応的なモデルとの比較を行う. PonderNet などの適応的なモデルは入力に応じて行う推論の深さを変化させる事により, 質問応答や多段推論など, 数量推論ではない他のタスクでは非適応的なモデルよりも高い精度を達成することが分かっている.

実験では, 多段の推論を要する推量推論タスクにおいて, 適応的な機構を持つ Ponder Transformer の振る舞いを検証し, 標準的な Transformer などの非適応的なモデルと比較を行う. データセットは形式言語によって構成され, 解答に要する推論の深さを明示的に定めることが可能となっている. また, 形式言語で書かれたデータセットは, DROP のような自然言語上の数量推論データセットから問題の構造のみ取り出したものと考えることができるため, 形式言語上でモデルの振る舞いを検証することは, 自然言語上での数量推論に対するモデルの振る舞いを検証する手助けになると考えられる.

本稿では, 現状で確認できている範囲の結果を共有し, 最後に冒頭の仮説を検証するための今後の方針を整理する. 実験の結果, 形式言語によって構成される数量推論データセットの中でも簡単な演算に対しての正解率は非適応的なモデルと適応的なモデルでは大きな差は見られなかったが, 入力に応じて適応的なモデルの層の深さが変化することが確認できた.

2 PonderNet

標準的なニューラルネットワークでは, 入力に応じて層の数が増減することは無い. 一方で, 一部では

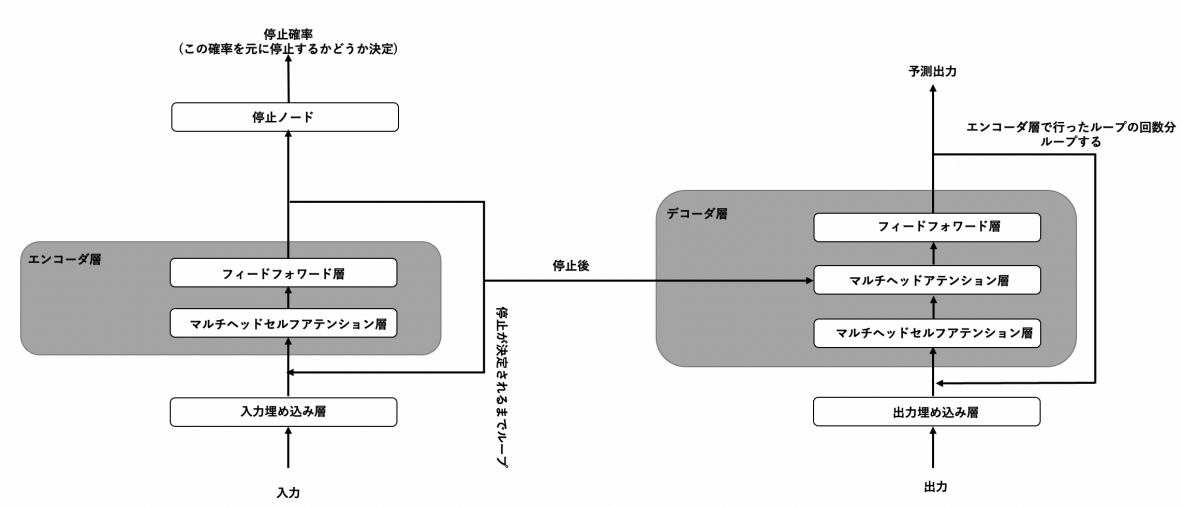

図 1　Ponder Transformer

そのような仕組みをモデルに組み込む研究が注目されており [8, 9], 本研究では中でも PonderNet [2] に着目する. PonderNet は入力に基づいて再帰回数を調整する事ができ, また, 既存のアーキテクチャに対しても簡単に適用可能なアルゴリズムである.

アーキテクチャ：　PonderNet の基本となるアーキテクチャは, 入力 x, 状態 h_n として, ステップ関数 s を用いて式 (1) のように表される.

$$y_n, h_n, \lambda_n = s(x, h_{n-1}) \tag{1}$$

式 (1) のステップ関数 s は, MLP [10] や LSTM [11] などの任意のニューラルネットワークや, Transformer の 1 層として考えることができる. また, このステップ関数 s は最大 N 回まで再帰的に適用され, 各 n 回目のモデルの出力が y_n となる. λ_n はこの n 回目の再帰的処理を停止する確率, h_n は h_0 を初期状態とした時の中間状態である. モデルは λ_n の値を元に停止するかどうかの決定を行い, 停止した際の y_n の値がモデルの出力となる.

損失関数：　損失関数は, ラベルを y として, 再構築項 L_{Rec} と正則化項 L_{Reg} で式 2 のように構成される.

$$\begin{aligned} L &= L_{Rec} + \beta L_{Reg} \\ &= \sum_{n=1}^{N} p_n \mathcal{L}(y, y_n) + \beta \mathrm{KL}(P || P_G(\cdot; \lambda_p)) \end{aligned} \tag{2}$$

p_n は λ_n によって求められる n ステップ目で初めて停止する確率 ($p_n = \lambda_n \prod_{j=1}^{n-1}(1 - \lambda_j)$), P は p_n の分布である (i.e., $P(n) = p_n$). また, L_{Rec} は, タスクで定義された損失関数 $\mathcal{L}$ の停止確率による期待値, L_{Reg} は P と式 3 のように定義された事前分布（N で切り捨てられた幾何分布, λ_p でパラメータ化されたもの）の間の KL ダイバージェンスである.

$$P_G(n; \lambda_p) = \lambda_p \prod_{j=1}^{n-1}(1 - \lambda_p) \tag{3}$$

3　人工数量推論データセット

本研究では, 事前学習用の簡単なデータセットとメインの多段推論用のデータセットの 2 つをプログラムによって人工的に生成し, 実験に用いる.

3.1　事前学習用データセット

多段推論で訓練する前にモデルが簡単な数量推論が可能になるように事前学習を行う. 図 2 上のように, 数値は 0 ~ 199 の範囲, 演算子は足し算「+」の 1 種類, 各式は 2 項演算によって構成されたデータセットを用いる. 事例の数は 40K である.

3.2　多段推論用データセット

各モデルの比較に用いるデータセットとしては, 図 2 下のようなデータセットを作成した. データセットは図のように数値の範囲が数値は 0 ~ 199, 演算子は足し算「+」の 1 種類, 変数は a ~ e の 5 種類で構成されている.

このデータでは, **式**は b=a+1 のような (最後の質問を除いた) 文脈中に出現するカンマ区切りの文字列と定義し, 「c=23+b，a=1，b=a+1，b=?」という事例の場合, 式の数は 3 である. また, 「**推論の深さ**」は, 「**解答に必要な式の数**」として定義する. 文脈中には解答に必要のない式もダミーの式として含まれてい

事前学習用データセット

```
Q: 25 + 32 =
A: 57
```

多段推論用データセット

```
推論の深さが1の場合
    Q: e=34, c=e+3, e=?
    A: 34

推論の深さが2の場合
    Q: b=12+a, d=b+3, c=11, a=c+67, a=?
    A: 78

推論の深さが3の場合
    Q: a=65+c, c=43+d, d=11, a=?
    A: 119
```

図 2　データセット概要図

るが, これらは問題を解くことに要する「推論の深さ」には影響しない. よって, 図 2 の深さ 2 の例は $c = 11 \rightarrow a = c + 67$ という推論過程を踏むため深さ 2 とみなされる.

検証にこのような形式言語による多段推量推論タスクを採用する強みは,

1. データの分布をコントロールする事ができる
2. 推論の深さなど問題の形式を自由に調整できる

の 2 点である. 1 に関しては, DNN モデルは, データセット中の分布の偏りに敏感であり, 本来の問題の構造に依らない解法を見つけてしまうことが知られている [12, 13]. 人工データを用いる事によりこの問題を極力排し, モデルの数量推論能力をより直接的に評価することができる. 2 に関しては, 推論の深さ, 数値の範囲やダミーの式の有無など, 検証に必要な要素を自由に制御可能なことにより, DNN モデルの数量推論に関する強みや限界を詳細に評価できる.

4　実験

4.1　モデル

以降の実験では以下のモデルを事前学習用データセット (§ 3.1) で事前学習し, 多段推論用データセット (§ 3.2) で追加学習・評価を行う.

Vanilla Transformer:　エンコーダ層, デコーダ層が共に 6 層ずつの標準的な Transformer.

Ponder Transformer:　PonderNet (§ 2) を Transformer 層に適用したものである. エンコーダ層, デコーダ層が共に 1 層ずつの図 1 のような Transformer の亜種であり, 図中の停止ノードの出力である停止確率を元にループの回数が決定する. 今回の実験では, エンコーダ層の [CLS] トークンに当たる出力を停止ノード (線形層) に入力し, 停止確率 λ_n を求める. また, 最大停止回数 N は 6, 損失関数の正則化項 L_{Reg} に対する重み β は $\{0.1, 0.01, 0.001\}$ を探索して最適な値, 幾何分布を決定するパラメータ λ_p は 1/4 とした.

Loop Transformer:　Ponder Transoformer と同様に, エンコーダ層, デコーダ層が共に 1 層ずつの Transformer であり, 同じ層に繰り返し表現ベクトル列を入力する (i.e., Vanilla Transformer の層間のパラメータを共有した亜種). Ponder Transformer と異なる点としてはループの回数が動的に変化せず, 6 回で固定されている点である.

各モデルに共通するハイパーパラメータとして, 最適化手法は Adam [14], 学習率は 0.0001, 埋め込みベクトルの大きさは 512, アテンション層のユニット数は 512, アテンションヘッドの数は 8 とした. また, 入力に対する埋め込みは文字単位で行い, タスクは 1 桁ずつの生成で解く.

4.2　多段の数量推論

学習・評価セット中に含まれる推論の深さ (§ 3) = 1, 2, 3 の事例の割合を変化させながらモデルの性能を評価した (表 1). 各設定ごとにデータセット全体を生成したのち, 学習・訓練データの割合が 8:2 となるようにランダムに分割した. また, ランダムにダミーの推論ステップを使う事で各事例中の式の数は 1 ~ 4 の範囲になるように設計する.

Ponder Transformer と Vanilla, Loop Transformer を比較すると, 精度による違いはあまり見られなかった. これは簡単な足し算のみのデータ上での比較であるためと考えられ, 今後更に複雑な問題上でも比較を行っていく必要がある. しかしながら, 推論の深さ 1 のものが解ける段階から深さ 2 の問題が解けるようになるためには, 実験データに含まれる推論の深さ 2 の事例数が 5K では不十分だった (表 3 行目) のに対し, 推論の深さが 3 の事例を解くことができるためには, データ中に推論の深さ 3 の事例を 5K 含めるだけで正解率がほぼ 1.0 になるという興味深い現象が見られた. ここから, 推論の深さ 2 の問題の解き方

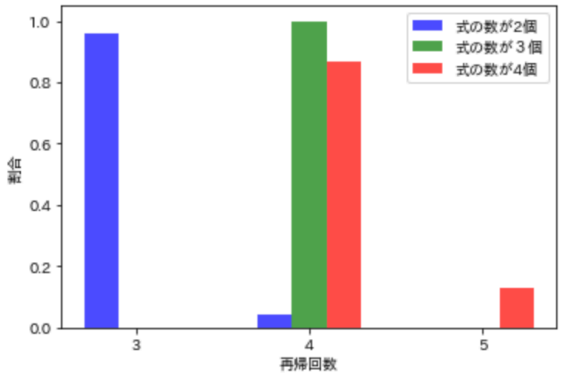

図 3　解答に必要な式の数が 2, 文脈中に出現する式の数が 2 ~ 4 の事例に対する Ponder Transformer の再帰回数のヒストグラム

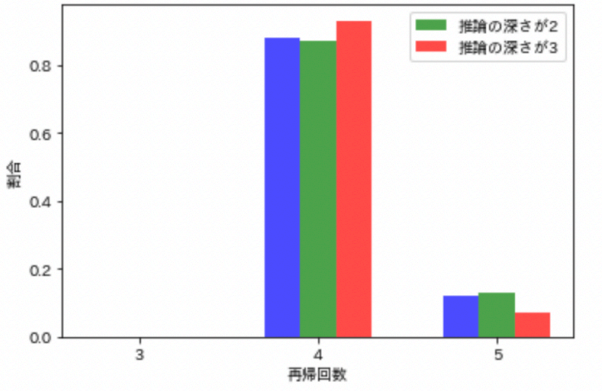

図 4　文脈中に出現する式の数が 4, 解答に必要な式が 1 ~ 3 の事例に対する Ponder Transformer の再帰回数のヒストグラム

の知識があれば, 深さ 3 の問題に対しては少量の事例のみで足りることが見て取れる.

表 1　推論の深さ (解答に必要な式の数) の割合を変化させた場合の正解率. x/y/z は推論の深さが 1/2/3 の事例が学習・評価データに合計して x,y,z 個あることを表している

推論の深さが 1/2/3 の事例の数	正解率		
	Vanilla	Loop	Ponder
0.5K/0/0	0.63	0.75	0.94
1K/0/0	0.945	1.0	0.985
1K/5K/0	0.3525	0.3658	0.4442
1K/15K/0	0.9134	1.0	0.9994
1K/15K/5K	-	0.9881	0.9879

4.3　Ponder Transformer の再帰回数に関する分析

以下の実験では, 再度推論の深さ 1, 2, 3 の事例をそれぞれ 1K, 15K, 15K 個用意し, 学習・訓練データに 8:2 でランダムに分割しモデルを訓練した. 学習させたモデルに対して, 推論の深さが 2, 式の数が 2 ~ 4 の事例に対する再帰回数の変化を調査した. 具体例として, 「a=1，b=a+1，b=?」は推論の深さが 2, 式の数が 2 の事例であり, ダミーの式を加えることが式の数の増加に対応する.

実験の結果, 図 3 のように, ダミーの式が増えるにつれてモデルの再帰回数も増加し, ダミーの式の有無にも依存してモデルが計算を行なっている事が分かった. これは, 今回構築したデータセットが文脈中に出現する変数の中からランダムにその値を問う形式になっていることに起因していると考えられる. 例えば, 「c=23+b a=1，b=a+1，b=?」のような入力に対しては, b の値だけではなく文脈中に出現するどの変数の値に対してもモデルが回答可能なように a, c の値も同時に計算していると仮説が考えられる.

また, 各事例中の式は, 他の式に出現していない変数に対し演算を行い, 鎖状に多段の数量推論を表現しているため, 推論鎖の長さがモデルの再帰回数の変化に反映されたということが図から観察される.

続いて, 文脈中に出現する式の数を固定して推論の深さを変化させた場合に対する再帰回数を調査した. 結果は図 4 のように, 解答に必要な式の数とは直接的に関係なく再帰回数が決定した. これは上記で述べたように, 答えるのに必要な式だけを見ているのではなく, 全ての変数の値を計算しているという仮説と一致する結果である.

5　おわりに

本稿では, PonderNet が数量推論においてどのように作用するのかを人工的なデータセットを用いて検証した. その結果, 簡単な演算に対する正解率は非適応的/適応的なモデル間では大きな差は見られなかったが, PonderNet が問題の構造に応じて再帰回数を変化させることが確認できた. この挙動に対する説明としての仮説を実験の節では提示したが, 仮説に対する厳密な評価は今後の課題である.

今回の実験は簡単な足し算のみのデータセット上での比較であったが, 我々は今後人工数量推論問題を用いてさらなる調査を行う予定である. 将来的には数量推論において, 適応的なモデルを用いて簡単な演算に対しては推論過程を短く行うように学習し, 複雑な演算に対してはより長い推論を行うことによる汎化性の向上を目指している.

謝辞

本研究は JST CREST JPMJCR20D2 及び JSPS 科研費 JP20J22697, 20K23314 の助成を受けたものです.

参考文献

[1] Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N. Gomez, Lukasz Kaiser, and Illia Polosukhin. Attention is all you need, 2017.

[2] Andrea Banino, Jan Balaguer, and Charles Blundell. Pondernet: Learning to ponder, 2021.

[3] Avijit Thawani, Jay Pujara, Pedro A. Szekely, and Filip Ilievski. Representing numbers in nlp: a survey and a vision, 2021.

[4] Dheeru Dua, Yizhong Wang, Pradeep Dasigi, Gabriel Stanovsky, Sameer Singh, and Matt Gardner. Drop: A reading comprehension benchmark requiring discrete reasoning over paragraphs, 2019.

[5] Mor Geva, Ankit Gupta, and Jonathan Berant. Injecting numerical reasoning skills into language models, 2020.

[6] Zhiyu Chen, Wenhu Chen, Charese Smiley, Sameena Shah, Iana Borova, Dylan Langdon, Reema Moussa, Matt Beane, Ting-Hao Huang, Bryan Routledge, and William Yang Wang. Finqa: A dataset of numerical reasoning over financial data, 2021.

[7] Eric Wallace, Yizhong Wang, Sujian Li, Sameer Singh, and Matt Gardner. Do NLP models know numbers? probing numeracy in embeddings. In **Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing (EMNLP-IJCNLP)**, pp. 5307–5315, Hong Kong, China, November 2019. Association for Computational Linguistics.

[8] Alex Graves. Adaptive computation time for recurrent neural networks, 2017.

[9] Mostafa Dehghani, Stephan Gouws, Oriol Vinyals, Jakob Uszkoreit, and Lukasz Kaiser. Universal transformers. In **7th International Conference on Learning Representations, ICLR 2019, New Orleans, LA, USA, May 6-9, 2019**. OpenReview.net, 2019.

[10] David E. Rumelhart and James L. McClelland. **Learning Internal Representations by Error Propagation**, pp. 318–362. 1987.

[11] Sepp Hochreiter and Jürgen Schmidhuber. Long Short-Term Memory. **Neural Computation**, Vol. 9, No. 8, pp. 1735–1780, 11 1997.

[12] Suchin Gururangan, Swabha Swayamdipta, Omer Levy, Roy Schwartz, Samuel Bowman, and Noah A. Smith. Annotation artifacts in natural language inference data. In **Proceedings of the 2018 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 2 (Short Papers)**, pp. 107–112, New Orleans, Louisiana, June 2018. Association for Computational Linguistics.

[13] Saku Sugawara, Kentaro Inui, Satoshi Sekine, and Akiko Aizawa. What makes reading comprehension questions easier? In **Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing**, pp. 4208–4219, Brussels, Belgium, October-November 2018. Association for Computational Linguistics.

[14] Diederik P. Kingma and Jimmy Ba. Adam: A method for stochastic optimization, 2017.